Sequential Circuits

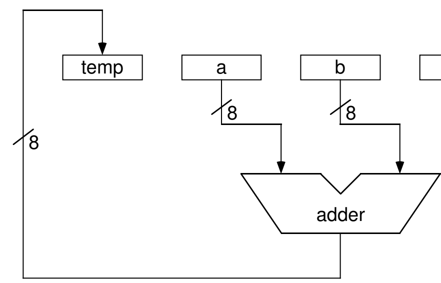
- For several lectures, we have studied *combinational circuits* — circuits whose output values depend *only* on the inputs
- Now we will study sequential circuits — circuits whose output values depend on *both*:
  - the inputs, and
  - the current *state* of the circuit
- The *state* of the circuit is defined by the values stored in the memory elements

---

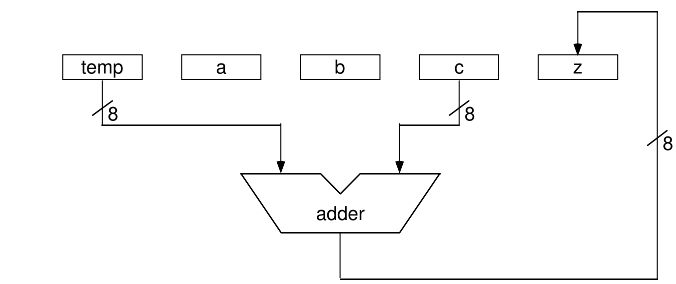
- With sequential circuits, we can build registers, memories, shift registers, counters, stacks, queues, etc.
- With combinational circuits, we can build adders, ALUs, and glue logic

A Very Simple Computer Datapath

- temp = a + b (a to left input…)

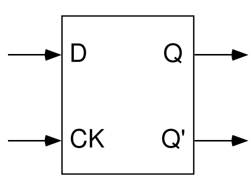


- z = temp + c (temp to left input…)



The D Latch (Gated D Flip-Flop)

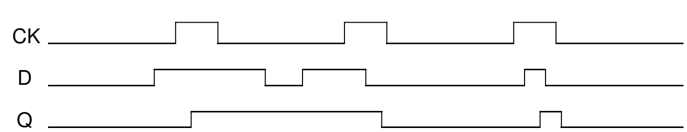
- Inputs:
  - D = data
  - CK = clock



- Outputs:
  - Q
  - Q'
- Operation:
  - When CK is 1, Q follows D (if D changes, Q changes also, after a short delay)
  - Otherwise, Q keeps its last value
- Potential problem: output is unstable anytime CK is 1
  - Useful for holding values to transfer outside the circuit
  - Not useful as a register

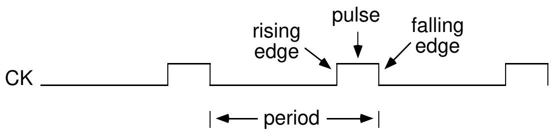
Timing Diagrams

- A *timing diagram* is a graphical representation of the signals in a circuit
  - The horizontal axis represents time
  - Vertical axis represents value of each signal:
    - A 1 is represented as a high line
    - A 0 is represented as a low line
  - The timing diagram may account for *propagation delay* — the delay through a combinational element (the amount of time it takes to execute)
- Timing diagram illustrating the operation of a D latch:



Clock

- It is common for a digital system to have a *clock* signal, which repeats a 0-1 pattern at regular intervals

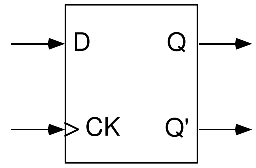


  - The *period* is the length of time that it takes to repeat the pattern
  - The part of the pattern which has the value of 1 is called the clock *pulse*
  - The 0-1 transition is called the *rising* (or *positive*) *edge*
  - The 1-0 transition is called the *falling* (or *trailing*, or *negative*) *edge*

- Clock signals are used in *synchronous* circuits — those where events happen at predictable, regular intervals

The (Edge-Triggered) D Flip-Flop

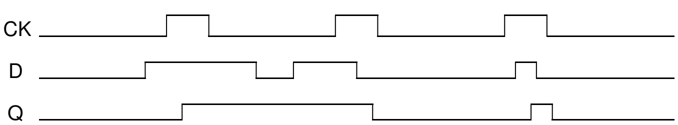
- Inputs:
  - D = data
  - CK = clock



- Outputs:
  - Q
  - Q'
- Operation:
  - When CK transitions from 0 to 1 (leading edge), Q is set to the value of D (after a short delay)
  - Otherwise, Q keeps its last value, regardless of any changes in D
- We can use *n* D flip-flops to build an *n*-bit register…

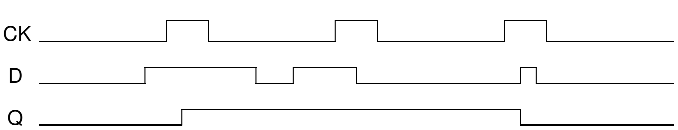
D Latch vs. D Flip-Flop

- D Latch:
  - When CK is 1, Q follows D (if D changes, Q changes also, after a short delay)



  - Not useful in building registers

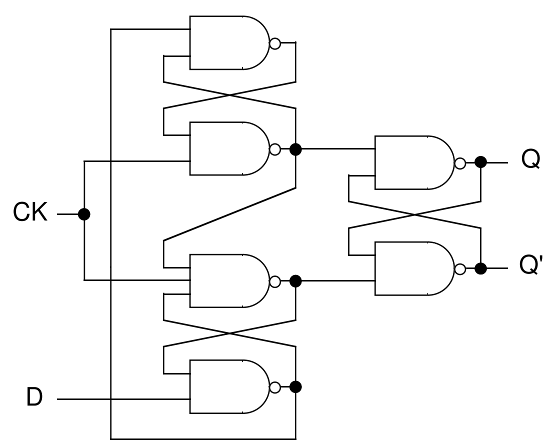
- D Flip-Flop:
  - When CK transitions from 0 to 1 (leading edge), Q is set to the value of D (after a short delay)



  - Useful as a memory element in a register

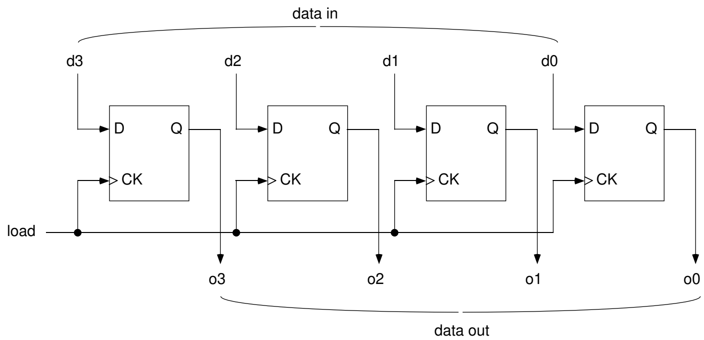
Building a D Flip-Flop

- Implementation is fairly complicated to analyze…



Building a Register Out of D FFs

- This simplified diagram shows how a 4-bit register can be constructed out of 4 D flip-flops



- When the *load* signal goes high, the *data in* values (d3–d0) are stored in the register, and become available as *data out* (o3–o0) signals

4 Major Types of Flip-Flops

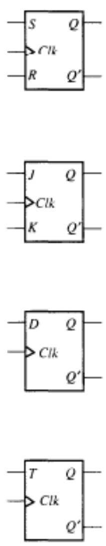


TABLE 6.1
Flip-Flop Types

FLIP-FLOP NAME	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR	S R | Q(next) 0 0 | Q 0 1 | 0 1 0 | 1 1 1 | NA	$Q(next) = S + R'Q$ $SR = 0$	Q Q(next) | S R 0 0 | 0 X 0 1 | 1 0 1 0 | 0 1 1 1 | X 0
JK	J K | Q(next) 0 0 | Q 0 1 | 0 1 0 | 1 1 1 | Q'	$Q(next) = JQ' + K'Q$	Q Q(next) | J K 0 0 | 0 X 0 1 | 1 X 1 0 | X 1 1 1 | X 0
D	D | Q(next) 0 | 0 1 | 1	$Q(next) = D$	Q Q(next) | D 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 1
T	T | Q(next) 0 | Q 1 | Q'	$Q(next) = TQ' + T'Q$	Q Q(next) | T 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Figure from *Principles of Digital Design*, Gajski, Prentice Hall, 1997